

General Network Performance Testing Methodology

Philip Joung

Application Note 02

July, 2003



Spirent Communications

26750 Agoura Road
Calabasas Hills, CA
91302 USA
e: enterprise@spirentcom.com

Sales Contacts:

North America

+1 800-927-2660

Europe, Middle East, Africa

+33-1-6137-2250

Asia Pacific

+852-2166-8382

All Other Regions

+1 818-676-2683

www.spirentcom.com/enterprise

Copyright 2003 by Spirent Communications, Inc. All rights reserved.

Table of Contents

1 Introduction.....	1
2 Why Test?	1
3 Methodology	2
3.1 A Stepped Approach	3
3.2 Consider Protocols and Applications.....	4
3.3 Consider Realism	5
3.3.1 User Realism	6
3.3.2 Network Realism	7
3.4 Applied Loads	7
3.4.1 Users	7
3.4.2 Connections.....	8
3.4.3 Transactions.....	8
3.4.4 Pages.....	8
3.4.5 Bandwidth	9
3.4.6 Relationships between Metrics	9
3.5 General Tests	9
3.5.1 Workload Validation	11
3.5.2 Maximum Connection Rate Test.....	12
3.5.3 Maximum Connections Test	13
3.5.4 Maximum Bandwidth Test	14
3.5.5 Maximum Simultaneous Users Test.....	15
3.6 Stress Tests	16
3.6.2 Long-Term Stability Test.....	17
3.6.3 Failover Test	18
4 Appendix	19
4.1 Test Phases.....	19
4.2 Things to Look for During System Failures	19
4.3 Spirent Communications Products	20
4.3.1 Avalanche	20
4.3.2 Reflector	20
5 Acknowledgements.....	20
6 References.....	21

1 Introduction

“There are many methods for predicting the future. For example, you can read horoscopes, tea leaves, tarot cards, or crystal balls. Collectively, these methods are known as ‘nutty methods’. Or you can put well-researched facts into sophisticated computer models, more commonly referred to as ‘a complete waste of time.’”

Scott Adams (1957 -), The Dilbert Future

“A machine is characterized by sustained, autonomous action. It is set up by human hands and then is more or less set loose from human control. It is designed to come between man and nature, to affect the natural world without requiring or indeed allowing humans to come into contact with it. Such is the clock, which abstracts the measurement of time from the sun and the stars: such is the steam engine, which turns coal into power to move ships or pump water without the intervention of human muscles. A tool, unlike a machine, is not self-sufficient or autonomous in action. It requires the skill of a craftsman and, when handled with skill, permits him to reshape the world in his way.”

J. David Bolter (1951 -), Turing’s Man: Western Culture in the Computer Age

Billions of dollars have been spent to make the Internet into the useful, often-indispensable tool used worldwide. It continues to pervade our daily life, integrating into areas that we often don’t even realize—picking up the phone at your office may very well route your call across the Internet. However, with all this value on the Internet, with all this investment and all this effort, the Internet (and other large networks) continues to face performance and reliability problems.

What can be done to help address these issues? Testing provides one of the answers to improving the state of communication and productivity on large networks such as the Internet. This document describes suitable methods and techniques that can help determine the performance and reliability of the systems on any network, large or small.

2 Why Test?

Why is testing so important? Without testing, only a few methods might help with deployment decisions: marketing literature, word of mouth, and personal experience. Marketing literature obviously focuses on showing the product in the best light—performance numbers usually show results under artificial conditions and best-case scenarios. Word of mouth (which includes magazine reviews and benchmarks), often has suggestions based on particular scenarios and experiences, many of which will not pertain to your configuration. Finally, the rapid and constant evolution of networks, protocols, applications and end-users they support makes it hard to rely on just personal experience to make critical network design and deployment decisions. Personal experience does indeed serve as a powerful tool to help improve the quality and effectiveness with which testing can be conducted.

The costs of network technology and implementations also continue to increase, and testing can actually help reduce costs. When deploying systems, the costs of under or over-provisioning are costly—too little equipment and performance, and the business productivity will have adverse effects; too much equipment, and the ROI of the deployment may never be realized, or may be extremely poor. Testing helps measure the actual performance of the deployment, helping to find bottlenecks,

increase performance, and deploy a system that properly meets a company's goals (high ROI, few errors, productivity enhancements, etc.).

Because of the proliferation of networks into daily use, the nonstop functioning of these networks has moved beyond simple user expectation to an often critical system, sometimes with life and death consequences. A 911 dispatch system, a hospital's medical information network, and an airport's air traffic control network all rely on the uninterrupted flow of information, with devastating consequences during outages. Other network outages may not be as dire, but the financial impact and decrease in reputation can cause significant pain as well. Testing provides the answer to validating and ensuring highly performing and available systems.

Many other advantages exist in testing, but they are beyond the scope of this paper. See Spirent's white paper *TR-1: High Volume Web Site Capacity Assessment* for a discussion on capacity assessment and the value that testing can have on the performance, quality and scalability of networks. Although the paper generalizes on Web sites, it contains points pertinent to anyone working with networks.

3 Methodology

The methodology in this document will help to understand the general performance characteristics of the system you plan to test. Most of the tests here will apply to many different situations, including testing firewalls, proxy servers, caches, and Web sites. There are companion methodology papers from Spirent that will provide guidance towards effectively assessing these families of devices. Along with these specific companion papers, a good paper to read for more useful tips during performance testing is *TR-5: The Network Commandments: Ten Suggestions for Improving Your Network Performance*.

Finally, I welcome any further suggestions for suitable ways that this methodology could be revised, improved and extended. This paper is for you, the testing community, and hence should reflect your ideas, thoughts and opinions. You should also adapt this methodology to your own situation, rather than just following the steps, as that will ensure the most effective and useful results.

"A doctor receives a call from his patient, who says, 'One month has passed since I saw you and I'm still not feeling well.' The doctor replies, 'Did you follow the directions printed on the medicine bottle?' 'Yes, I did', said the patient, 'It says 'Keep tightly closed.'"

Some assumptions for this methodology:

- You have a good understanding of the network and the applications on the network.
- You understand how to maintain and configure the system(s) being tested.
- You have access to the system(s) before deployment into the live environment (testing will cause many systems to fail—an important finding in itself).
- You have the ability to generate traffic that realistically models the major protocols and applications on your network.
- Sufficient traffic can be generated to exceed the peak traffic levels expected by your network.

- For device-in-the-middle testing, you have the ability to generate and respond to realistic traffic (e.g. both a client and a server simulator) with sufficient volumes to adequately assess the device.

3.1 A Stepped Approach

A testing methodology can be described as a series of steps. By doing so, the impact and timing of each step becomes easier to predict, helping to reduce the likelihood of time and cost overruns. Figure 1 depicts some suggested steps in a system testing methodology.

After the initial planning phase, thorough testing involves a feedback process that improves the system under test through the course of several iterative tests.

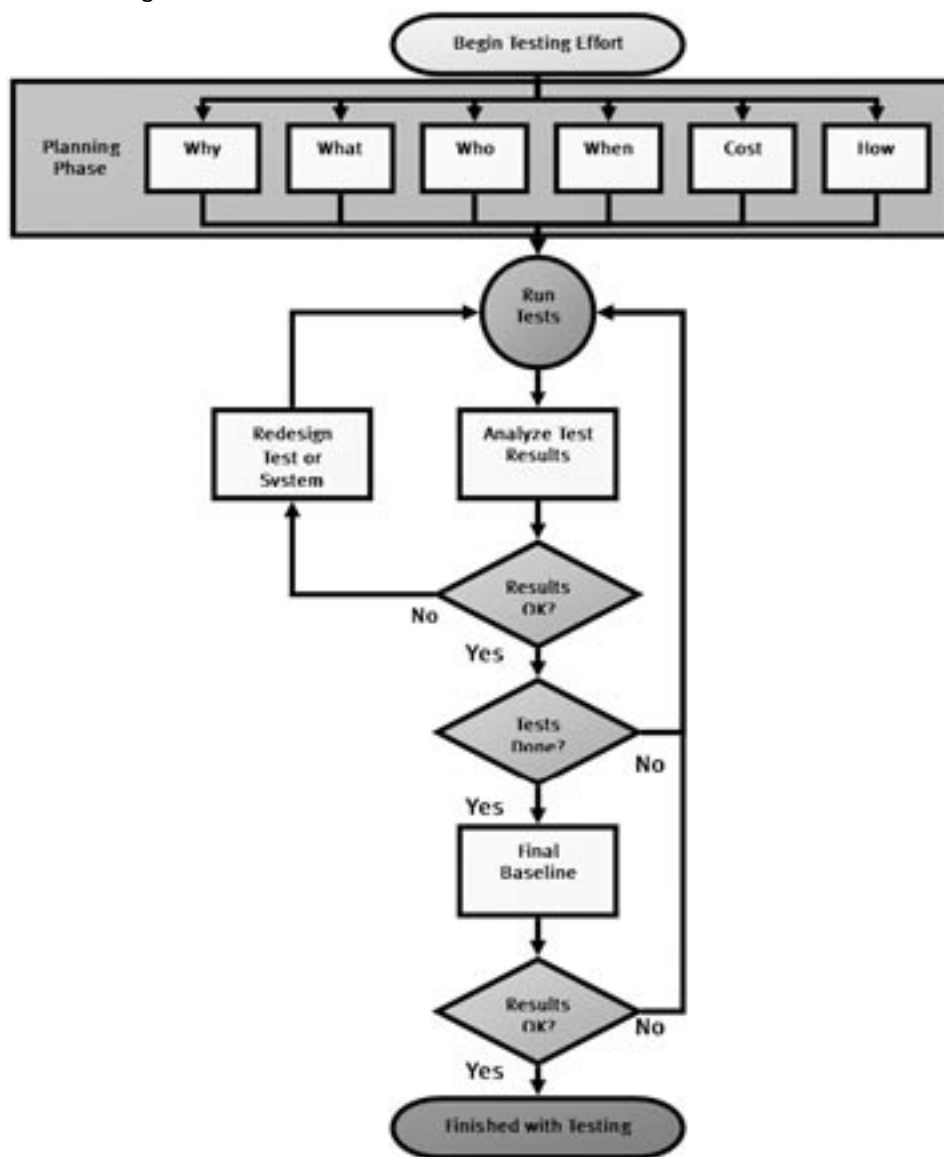


Figure 1: A simplified flow chart of the steps in testing a system.

A good methodology always starts with planning. As with any complex project, a little extra effort at the beginning will pay off with vastly improved results on the back end. Things to consider during the planning phase include:

- **Why:** Document the goals of testing, and look for how they might map into the overall business objectives and get buy-in from management. Determine the purpose of the system being tested, and document what contributions a testing methodology might have towards this.
- **Who:** Ensure enough people are available to conduct this assessment, and that they have the proper skills and training. Obtain management agreement with the objectives of this testing (along with the benefits above) and the costs that will be incurred.
- **What:** Determine what should be tested (see section 3.2 for some tips). Look for tools that can properly address the requirements and goals of the test. Consider what the expected results will be along with what steps to take to achieve those results if testing does not initially show this. Define the success of the testing and application functionality and RASSP in both qualitative and quantitative terms.
- **When:** Plan the amount of time that testing will take and incorporate this time into the overall product delivery plan. Work with the development team to conduct tests early on and to have them facilitate testing efforts through designs and features that enhance visibility during testing.
- **Cost:** Consider and document the anticipated costs for testing, including any expertise that must be hired or trained and tools and equipment that must be acquired.
- **How:** Create a testing methodology that not only determines performance and locates bottlenecks, but incorporates enough flexibility to evolve and respond to different factors and ideas.

After the planning step, the most effective testing incorporates an iterative feedback process: run the test, analyze the results, use the results to improve both the subsequent testing and/or the system(s) being tested, and run again. To test complex, poorly understood systems, a systematic approach of changing only one thing at a time between retests can help isolate particular issues and problems within the infrastructure. For example, after a test, analysis of the results shows that the database's lack of indexing is causing performance issues: one could index the database, increase the memory, and add extra hard drives before retesting, but the more useful scenario involves testing three times with each separate change to quantify the improvements, if any, of making each particular change.

After the iterations produce the desired results, take the time to conduct a final comprehensive performance analysis of the system before turning it live. This serves as a final "double-check", helping to reveal any unforeseen issues, and produces a final baseline result that helps during future comparisons of performance changes.

3.2 Consider Protocols and Applications

A network exists in order to share information and serve particular functions, whether it be Web browsing, an order entry system, email, downloading music, or networked game-play. These applications all use particular protocols that use particular parts of the network. When considering the protocols/applications to focus on, look at both the prominent protocols and the most important applications on your network. Certain applications, for example, may take up a large portion of the network traffic but have little business value, e.g. peer-to-peer music file sharing. Others may conversely take little network traffic and yet be considered critical to the operation of the network and/or the business (payroll applications, etc.). Carefully considering both of these situations should help create a protocol and application mix that will more realistically assess the performance and stability of the system and its ability to handle network traffic, especially the ones that really matter to the business.

After choosing the protocol(s), consider the mix of traffic that may make sense for the system and the network(s) it will function in. For HTTP, it may suffice to include a mix of HTTP 1.0 and HTTP 1.1, along with a few pages of differing sizes and content. If the system works with SSL (unlike some systems which simply pass SSL unaltered, like most firewalls and load balancers), then ensure that the testing incorporates SSL, as the results with SSL will almost certainly look different than without SSL. Streaming protocols typically have longer lasting connections, with connections at differing speeds. FTP may have very persistent connections, with long periods of both activity and inactivity.

For the purposes of this methodology, we will focus on the HTTP protocol, which continues its dominance as one of the major protocols on the Internet along with its particular importance and prevalence in most networks. Key advantages include its combination of both long and short-lived connections and well-developed application-level performance and data-integrity measurements. However, the techniques and ideas in this methodology pertain to many other protocols, and can be adapted with little effort to apply to other protocols.

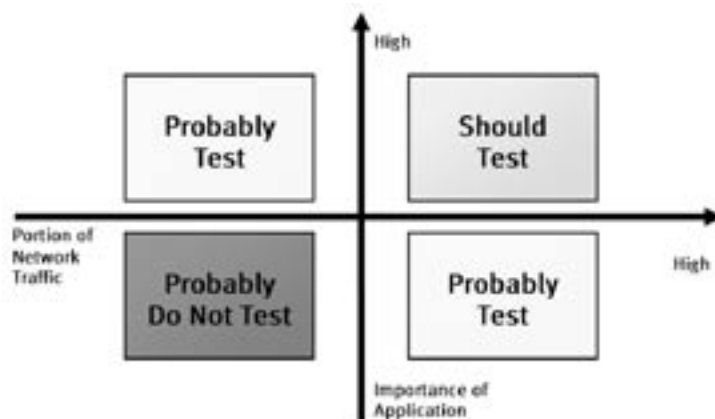


Figure 2: Choosing the protocols and applications to use during testing can make a difference in the validity of the results. Focus efforts on the most important applications and the ones that comprise the greatest portion of network traffic.

3.3 Consider Realism

Testing has many goals, including performance, reliability and security. The tests included above have the goal of determining maximum performance under the conditions that best achieve the result. For most networks, the information gleaned from these tests helps in capacity planning, but the workloads most likely do not reflect the traffic that will be experienced by a real network. To achieve a result that makes sense for a real network, testing must use realistic traffic.

Realism in test traffic falls into two major categories, user realism and network realism. Both parameters will have different effects on the network as well as different criticality during testing. However, by combining these two factors during testing, the results will more closely reflect the results when deployed live. For more in-depth discussion on realism, see *TR4: Reality Bytes: Why Realism Matters for Capacity Assessment*.

3.3.1 User Realism

Users often exhibit strange, erratic behaviors that can cause significant performance issues for the systems they use. Examples of this behavior include:

- **System usage:** Users will work the network in different ways—some may simply load a Web site home page and leave. Other users may go through most of the site to shop and search for items, add them to a shopping cart, enter credit card information and make the purchase, while at the same time download music, send and receive email, and view the news over a audiovisual stream. Usage behaviors can significantly affect the network systems, and capturing and simulating the representative behaviors will create better test results.
- **Think times:** The amount of time a user spends between system accesses. For example, a Web site visitor might load the home page, read the page for a few seconds before clicking on the next link. The time spent reading and deciding is called the think time. FTP users will often seek the next file to download—the time spent looking for the next download is considered think time as well.
- **Client application usage:** A great many applications exist for activities like browsing the Web, viewing streaming media, and conducting FTP downloads. Each of these applications has particular parameters and behaviors that often affect the systems that they work with. For example, Web browsers use specific versions of HTTP which affect network and system performance. Sites also will often tailor content towards specific browsers, which adds overhead that affects performance.
- **SSL usage:** While SSL increases the security of Web pages, its use comes with a heavy hit on performance. Performance decreases of 90% or more are not uncommon, and testing should include SSL if it will be used in the final system.
- **User frustration:** When users encounter a network that is slow, they will often abandon their download and either try again (usually not helpful, as it adds to the load) or simply leave (resulting in lost business and revenue). For example, when users hit the stop button on a Web browser, it will no longer accept any data that comes from the Web site. However, the site often continues processing the original request only to send the result back to a browser that will discard the result.
- **Cookies/Dynamic URLs/Session IDs:** Browsers connect and download information from a Web site using the HTTP protocol. In an attempt to allow a Web server to handle more users, the protocol was made stateless, meaning it has no notion of what user is connecting and downloading information from the site. In order to get around this

statelessness, several strategies have been devised: cookies, dynamic URLs and session IDs. In order to realistically simulate user behavior, testing should incorporate these user tracking mechanisms, which all add load to a system infrastructure.

3.3.2 Network Realism

Along with user behaviors, the network can exhibit issues of its own. The most problematic of these issues can result in dramatic performance losses, so testing with these parameters can be important for networks that experience these network issues.

- **Packet loss:** While several enhancements in TCP/IP networks have reduced the incidence of packet loss, it still remains a real issue (see <http://average.matrixnetsystems.com/> for statistics on the current amount of packet loss on various parts of the Internet). NASA, for example, published a study where FTP throughput dropped by 50% with only 3% packet loss.
- **Network latency:** Latencies cause significant performance issues as well—the slower network information enters and leaves a system, the more load the system must endure.
- **Jitter:** Systems that rely on fixed, dependable packet arrivals can experience significant problems when jitter increases. Voice and streaming applications, in particular, significantly degrade with increasing amounts of jitter.
- **IP fragmentation:** Streaming media has become one of the more prevalent sources of IP fragmentation [CAIDAO2]. While not problematic in small amounts, IP fragmentation often becomes a problem when facing large amounts of traffic and out-of-order arrivals. When combined with packet loss, the two can amplify network problems, as the loss of even 1 fragment results in the discard of all the other corresponding fragments.

3.4 Applied Loads

Network and computing devices all have particular values that matter most when it comes to performance. For example, A Web site might want to know how many simultaneous users it could handle during peaks, and a firewall manufacturer might want to determine the maximum amount of bandwidth that it can handle. When testing, the test tool can generate traffic that focuses on a particular metric, trying, for example, to see if the system under test can effectively handle a certain number of simultaneous users. Choosing the right load to apply will ensure that the system being tested gets assessed in the proper way and that the test produces results that matter for the system under test.

3.4.1 Users

The user handling capacity of a system often stands out as one of the more important metrics. Indeed, networks ultimately exist to serve the needs of its users, so knowing that a network will properly handle the desired level of user traffic is critical. For the purposes of testing, a user equates to the simulation of a person using the network. For example, a user could be configured to browse a Web

site using the Netscape Navigator browser in order to search for and purchase some items from the Web site, while at the same time read some email, download some favorite MP3 songs, and catch up on the news using streaming media. Or, a user could be configured to simply load one HTML page and go away. Two pertinent metrics exist for users: maximum simultaneous users and maximum sustainable user arrival rate (users per second). Systems that use this metric include Web caches, Web servers and server load balancers.

3.4.2 Connections

The networking world loosely defines a connection as a channel between two systems for transferring information. Depending on the system, protocol and applications being used, network connections can be short-lived or long, persistent connections. The HTTP protocol has two major versions currently in use, HTTP 1.0 and HTTP 1.1, with the major difference between the two being the persistence of the connection¹—HTTP 1.1 has the ability to initiate a persistent connection between client and server, using the same connection to transfer multiple objects. This reduces the amount of overhead devoted to connection setup and teardown. Do not confuse the protocol-level persistence of HTTP 1.1 with application-level persistence, which uses methods such as cookies and session IDs to maintain user persistence. Other potentially persistent connections include FTP, SMTP and POP3.

There are occasions when a standard connection does not exist, such as in the case of streaming protocols, DNS and SNMP, which run over the connectionless UDP protocol. In this case, the source sends its message with the hope, but no guarantee, that the message will reach its destination.

As with users, two major metrics pertain to connections: maximum simultaneous connections and maximum connections per second. Devices such as firewalls and intrusion detection systems use maximum simultaneous connections and connections per second as key metrics.

3.4.3 Transactions

On networks, a transaction is defined as the initiation and successful completion of a particular communication request. Examples of transactions include the downloading of one MP3 file using FTP, sending one email with SMTP, or downloading one graphic using HTTP. Browsing the Web and working with email (during which a user often sends and receives dozens of emails) are examples where a particular session usually involves several transactions. Taking the example of Web browsing, a particular Web page comprises several objects: the HTML page itself and the other objects that get loaded with the page, including graphics, java applets, cascading style sheets, and Flash animations. For the purposes of this methodology, consider every object that loads to be a single transaction. On the World Wide Web, this definition of a transaction equates to a hit, popularly used in the early days of the Web to describe site traffic and popularity.

Two major metrics exist for transactions: maximum total transactions and transactions per second. SSL accelerators, Web sites and Web caches use these metrics most often.

3.4.4 Pages

Web sites now use page views as the major traffic reporting metric, supplanting the original practice of quoting hits to denote the traffic encountered by the site. This makes sense, as it not only matches how the average user experiences a site, but it more accurately portrays traffic on a site for advertising purposes. The relevant metric for pages is maximum pages per second.

3.4.5 Bandwidth

The ability of a system to handle a particular amount of bandwidth can be important in capacity planning decisions. This often occurs in determining and planning for capacity of networks, trying to ensure that a particular system can handle the needed throughput for a network (e.g. ensuring that a firewall will adequately handle the traffic for a T3 connection to the Internet).

However, bandwidth often matters more in lower-level network systems such as routers and switches. When working with more intelligent network systems such as stateful firewalls, Web sites, intrusion detection systems and content switches, bandwidth is usually a by-product of the other applied loads above—more transactions or users mean more bandwidth utilization. During testing, look for bandwidth utilization while applying one of the above loads and ensure that it stays below the bandwidth amounts available in the production network (e.g. if the bandwidth to the Internet is a T3, make sure during testing that the bandwidth with realistic loads remains under 45 Mbps). Finally, the goals of the system/application and of the testing itself will determine whether an applied amount of bandwidth becomes an important consideration versus the other metrics above.

3.4.6 Relationships between Metrics

The above metrics have a loose relationship to each other—how much depends on factors such as the

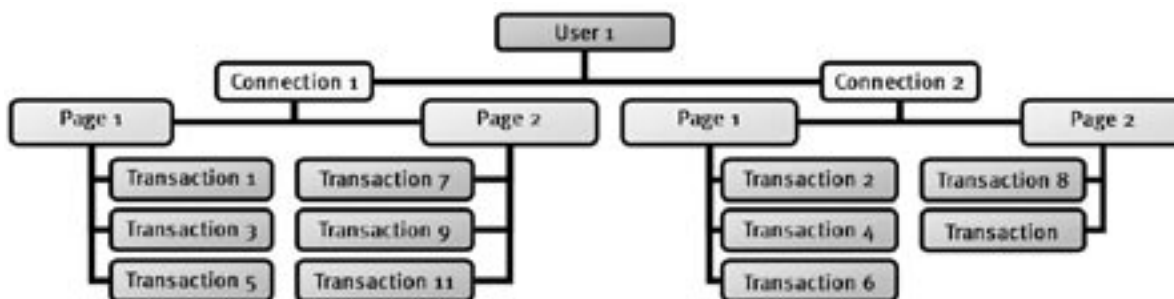


Figure 3: A depiction of the typical relationship between users, connections, pages and transactions on a Web site.

protocol(s) and settings used on the system along with the behavior of the users. Take as an example the HTTP protocol. The only time that users, connections, transactions and pages are equal is when each user downloads only one HTML page with no graphics or other objects and closes out the session. More typically, a single user will use two or more connections to download several pages, generating many transactions per page. Bandwidth utilization will typically increase in proportion to the other metrics as well. Figure 3 depicts a typical relationship between the above metrics.

3.5 General Tests

What follows is a list of tests adaptable to assess the performance of most systems. The methodologies below are generic, allowing one to use a wide range of tools to conduct the assessments. Spirent Communications has a wide range of tools and solutions that help with this testing, including the Avalanche family of security, Web and application testing products. For more information on Spirent Avalanche, refer to section 4.3 of the Appendix. An explanation of test phases also appears in the Appendix.

Methodology Definitions

- **Result:** provide information about what the test will accomplish.
- **Purpose:** explains the value and focus of the test, along with some simple background information that might be helpful during testing.
- **Test specification:** the load to apply during the test, as explained in section 3.3.
- **Constraints:** details any constraints and values that should not be exceeded during testing.
- **Time estimate:** a rough estimate of the amount of time that the test may take to complete.
- **Type of workload:** in order to properly achieve the goals of the test, each test requires a certain type of workload. This methodology specification provides information on the appropriate script of pages or transactions for the user.
- **Methodology:** a list of suggested steps to take in order to assess the system under test.
- **What to look for:** contains information on behaviors, issues and errors to pay attention to during and after the test.

3.5.1 Workload Validation

Result	This test will validate workloads, ensuring proper functionality with minimal loads. Please note that this is not a test in itself—instead, every other methodology below uses this one test to validate the workload.
Purpose	This test helps ensure that the workload functions correctly, helping to discover connectivity issues, syntax errors and other problems that might occur before the high-load test is run. Use this validation first before running the other methodologies below.
Test specification	Users
Constraints	1 user (per protocol if applicable)
Time estimate	Specification: about 30 minutes Run time: about 5 minutes
Type of workload	Dependent on test to be run
Methodology	<ol style="list-style-type: none"> 1. Determine the workload to use, based on suggestions included in section 3.2 and/or the methodology being run. 2. Configure the tool to ramp up traffic to 1 user and hold steady for 240 seconds. 3. If the testing tool supports this functionality, have the test tool validate the test first and check for any errors. Some test tools also generate a PCAP file during validation, making network troubleshooting much easier should problems arise. Resources are available on the Web to load and interpret PCAP files. 4. Run the test. 5. During the test, use the tool's real-time reporting interface to look for errors during the test. 6. After the test, analyze the detailed statistics to locate errors and particular system issues. 7. Of note are application level errors and connectivity errors. For example, this test may show that a connection error exists in the test network. It may also find typos or unintentional errors in the workload (e.g. HTTP 404- page not found errors). 8. Correct or resolve any errors encountered. 9. Rerun the test until no more unexpected errors exist.
What to look for	<p>Application level errors</p> <p>Application timeouts</p> <p>Unexpected errors—e.g. page not found, wrong page returned</p> <p>Network level errors</p> <p>Network timeouts</p> <p>System configuration errors</p> <p>Unexpectedly long response times</p>

3.5.2 Maximum Connection Rate Test

Result	This test establishes the maximum number of connections per second the system will successfully handle.
Purpose	Having the effective connections/sec for the system helps provide an understanding of the rate of traffic that the system should handle and points to potential bottlenecks in the system.
Test specification	Connections per second
Constraints	None
Time estimate	Specification: about 10 minutes Run time: about 1 hour
Type of workload	Use a workload with small file sizes. During the test, the system under test should encounter a high rate of connections opening and closing, with fast retrievals of small objects. Keeping the file sizes small helps reduce the likelihood of hitting a bandwidth limitation.
Methodology	<ol style="list-style-type: none"> 1. If possible, obtain the expected handling rate of the system using the system’s specifications, denoted here by ExpMaxCR. If ExpMaxCR is not known, start with a “guesstimate” for ExpMaxCR and pay attention during the run in step 6 and 7 below to see if the system hits its maximum capacity. If not, then double the first guess for ExpMaxCR and run again. 2. Use a successful workload determined in the methodology listed in section 3.5.1. 3. Configure the test tool with relatively large step heights of (ExpMaxCR/10). Ramp up each step over a ramp time of 15-30 seconds to ensure the system has the chance to make a best effort at keeping up with the new load. 4. Configure enough steps to surpass ExpMaxCR by at least 50%, say 15-20 steps. 5. Step steady time for each step should be at least 120-240 seconds to maximize the chances that the system under test reaches steady state. 6. Run the test. 7. During the test, look to ensure that the system has a chance to reach “steady state” during the earlier steps where the traffic does not exceed ExpMaxCR. If it does not seem to hit steady state, rerun the test with a longer step steady time. 8. After this first test, find the level at which the system cannot keep up with the traffic and errors are encountered, which we’ll denote as MacroMaxCR. 9. Using the level MacroMaxCR determined at the previous step as the new median to ramp towards, configure a new test with smaller incremental steps. Configure a starting ramp up to (MacroMaxCR-20%) over 120 seconds and hold steady for 240 seconds. 10. Next, configure ramp heights of (MacroMaxCR/100) with a step steady time of 60 seconds each for a total of 40 steps and a ramp time of 10 seconds. 11. Run the test with the new parameters. 12. Afterwards, determine the maximum sustainable connections per second by finding the level at which traffic continues without errors.
What to look for	<p>Error conditions, as listed in the Appendix in section 4.2.</p> <p>Often, because of application level timeouts and TCP retransmissions, the actual load at failure is lower than what may be indicated. This is why it is important to look for signs that the system has reached steady state when each step is taken.</p>

3.5.3 Maximum Connections Test

Result	This test establishes the maximum total number of connections the system will successfully handle.
Purpose	Having the maximum connections for the system helps provide an understanding of the total number of connections that the system should handle and shows whether the system has a bottleneck.
Test specification	Connections
Constraints	Be sure that the connections/sec does not exceed the rate determined in the previous test (3.5.2). Configure the users to abort (click-away) any transaction that does not return valid traffic after 15 seconds.
Time estimate	Specification: about 10 minutes Run time: about 1 hour
Type of workload	Use, small objects, many persistent connections, and long user think times. For example, use HTTP 1.1 with persistence (on both clients and servers) and a 60 second think time. In this scenario, the simulated user will retrieve an object, hold the connection open for 60 seconds, retrieve the second object, and close the connection. This helps ensure that many connections are created and kept open for long periods.
Methodology	<ol style="list-style-type: none"> 1. Validate the workload based on steps in section 3.5.1. 2. If the system under test has an expected maximum simultaneous connections, use that number as a guide, denoted here as ExpMaxConn. If ExpMaxConn is not available, take a “guesstimate” for ExpMaxConn and continue with this methodology. During steps 5 and 6, look to see if the system hits its capacity. If not, double the initial guess for ExpMaxConn and try again. 3. Configure a test to ramp up connections with steps heights of ExpMaxConn/10 for 20 steps. Each step ramp time should be 15-30 seconds. 4. Configure the tool to hold each step for a sufficient amount of time to reach steady state, say about 120-240 seconds. At minimum, the step steady time should exceed the think time of the users. 5. Run the test. 6. During the test, look for errors and other failure conditions to appear. 7. Analyze the post-run statistics for behaviors and conditions that show that the system is running out of connection resources and obtain the corresponding open connections, denoted as MacroMaxConn. 8. Configure another test to ramp up to (MacroMaxConn-20%) over a span of 120 seconds and hold steady for 240 seconds. 9. After the ramp up, configure the tool with step heights of (MacroMaxConn/100) over 40 steps, holding each step for 60 seconds. 10. Run the test with the new parameters. 11. Afterwards, determine the maximum simultaneous connections by finding the level at which traffic continues without errors.
What to look for	Pay attention to the results to look for timeout errors. Given the nature of this test, do not overlook the fact that the connections, while being kept open, may in fact be stalled, passing no useful traffic. This is also the reason to have the test tool close the connection after the second GET, so that any stalled connections can be closed out and errors logged.

3.5.4 Maximum Bandwidth Test

Result	This test establishes the maximum bandwidth that the system will successfully handle.
Purpose	Having the maximum bandwidth for the system helps provide an understanding of the bandwidth that the system should handle and shows whether it contains a bottleneck. For deployments in which bandwidth and throughput are critical, it is important to know that the system will not cause performance problems.
Test specification	Bandwidth
Constraints	<p>Be sure that the connections/sec does not exceed the rate determined in test 3.5.2.</p> <p>Be sure that the total number of connections does not exceed the value determined in test 3.5.3.</p> <p>Ensure that the test tool and the connectivity devices used during the test do not become bottlenecks. For example, be sure that the switch fabric has enough capacity to exceed the bandwidth expected during the test.</p>
Time estimate	<p>Specification: about 10 minutes</p> <p>Run time: about 1 hour</p>
Type of workload	The average size of an HTTP transaction often falls in the range of 8-13 kilobytes. For tools that support dynamic file sizes during a test, use a test that increases the file size, thereby increasing bandwidth utilization. If not, have the tool increase load using connections/sec with file sizes between 8 and 13K. If the tool cannot create enough bandwidth with these average file sizes, then increase the file sizes.
Methodology	<ol style="list-style-type: none"> 1. Obtain the expected bandwidth capacity of the system from datasheets, denoted here as ExpBnd. If unavailable, take a guesstimate for ExpBnd. 2. Validate the workload using steps outlined in section 3.5.1. 3. Configure the test tool to ramp up the bandwidth in heights of ExpBnd/20, holding each step steady for 120 seconds, for a total of 30 steps. 4. Run the test. 5. During the test, check to see that steady state is achieved at each step. 6. If a guesstimate was used for ExpBnd, check to see if bandwidth utilization tops out. If not, double ExpBnd and start over from step 4. 7. Using the post-test results analysis, obtain the maximum bandwidth, denoted here by MacroEffBnd. 8. Configure another test to ramp up bandwidth to (MacroEffBnd-20%) over a span of 120 seconds and hold steady for 240 seconds. 9. After that, increase the bandwidth in steps of (MacroEffBnd/100) for 40 more steps, holding each step for 60 seconds. 10. Run the test with the new parameters. 11. Analyze the post-test results to obtain the maximum effective bandwidth.
What to look for	As bandwidth utilization increases, the resources of the system under test will decrease. Eventually, errors and timeouts will likely occur when the system resources become scarce.

3.5.5 Maximum Simultaneous Users Test

Result	This test establishes the maximum total number of users the system will successfully handle.
Purpose	Having the maximum users for the system helps provide an understanding of the total number of users that the system should handle and shows whether the system has a bottleneck.
Test specification	Users
Constraints	Be sure that the connections/sec does not exceed the rate determined in test 3.5.2. Do not allow the maximum connections to exceed the amount determined in the maximum connections test (3.5.3). Configure the users to abort (click-away) any transaction that does not return valid traffic after 15 seconds.
Time estimate	Specification: about 10 minutes Run time: about 1 hour
Type of workload	Use, small objects, many persistent connections, and long user think times. For example, use HTTP 1.1 with persistence (on both clients and servers) and a 30 second think time. In this scenario, the simulated user will retrieve an object, hold the connection open for 30 seconds, retrieve the second object, and close the connection. This helps ensure that many users get created and stay around for long periods.
Methodology	<ol style="list-style-type: none"> 1. Validate the workload using steps outlined in section 3.5.1. 2. If the system under test has an expected maximum simultaneous users, use that number as a guide, denoted here as ExpMaxU. If ExpMaxU is not available, take a “guesstimate” for ExpMaxU and continue with this methodology. If maximum number of connections is available, divide that number in half to use as the guesstimate. During step 6, look to see if the system hits its capacity. If not, double the initial value for ExpMaxU and try again. 3. Configure a test to ramp up connections with step heights of ExpMaxU/10 for 20 steps. Each step ramp time should be 15-30 seconds. 4. Configure the tool to hold each step steady for a sufficient amount of time to reach steady state, say about 120 seconds. At minimum, the step steady time should exceed the think time of the users. 5. Run the test. 6. During the test, look for errors and other failure conditions to appear. 7. Analyze the post-run statistics for behaviors and conditions that show that the system is running out of connection resources and obtain the corresponding simultaneous users, denoted as MacroMaxU. 8. Configure another test to ramp up to (MacroMaxU-20%) over a span of 120 seconds and hold for 240 seconds. 9. After the initial ramp up, configure the tool with step heights of (MacroMaxU/100) over 40 steps, holding each step for 60 seconds. 10. Run the test with the new parameters. 11. Afterwards, determine the maximum simultaneous users by finding the level at which traffic continues without errors.
What to look for	Pay attention to the results to look for timeout errors. Given the nature of this test, look that the connections, while being kept open, may in fact be stalled, passing no useful traffic. This is also the reason to have the test tool close the connection after the second GET, so that any stalled connections can be closed out and errors logged.

3.6 Stress Tests

By its very definition, stress connotes an abnormal condition. When conducting stress tests, we hope to push the system beyond its “normal” operating conditions in order to glean valuable insights into system behavior, performance and stability at these high loads. For Web sites, traffic surges often exceed 10-20 times the normal level of traffic—if a site will encounter these surges in traffic, surge tests will help confirm the ability of the site to properly deal with this onslaught of traffic.

3.6.1 Traffic Surge Test

Result	This test determines the ability of the system to handle large surges in traffic.
Purpose	Systems can often encounter large traffic jumps, whether planned or unplanned. Examples include surges from a television advertisement, tax filing deadlines, and last-minute holiday purchases. These surges often correspond with times where outages can be extremely costly. For example, during a last-minute holiday purchase rush, any outages will cost more because more is being purchased during that time and the likelihood of defection during a failure is also higher. This test helps determine the system under test’s ability to manage and deal with these surges.
Test specification	System dependent: choose the specification that matters most. See section 3.3 for more information.
Constraints	Be sure that the connections/sec does not exceed the rate determined in test 3.5.2. Be sure that the total number of connections does not exceed the value determined in test 3.5.3. Keep bandwidth utilization below the amount determined in test 3.5.4.
Time estimate	Specification: about 30 minutes Run time: varies from minutes to about 1 hour
Type of workload	In this test, use a workload that represents what might be expected in a production network. Section 3.2 has information to help with this.
Methodology	<ol style="list-style-type: none"> 1. Validate the workload using steps outlined in section 3.5.1. 2. Choose load specification to apply and use the maximum performance obtained in section 3.5, denoted here by ExpMaxPerf. 3. Based on think times, download times and other performance issues, determine the maximum time a workload will take to complete, denoted here by WorkTime. 4. Configure the test tool to ramp up the load in heights of ExpMaxPerf/50, holding each step steady for WorkTime*2 seconds, for 50 steps, and ramp times of 15 seconds. 5. Run the test. 6. Analyze the results, looking for performance issues and bottlenecks. 7. Resolve any troublesome issues and repeat this test.
What to look for	<p>Unlike other tests, steady state may never be reached during each step, especially with complex workloads. The purpose is to determine behavior under surge conditions, not steady-state conditions.</p> <p>Timeouts, large numbers of open connections, and unexpected errors usually point to an overloaded system.</p> <p>The system under test may never complete this test without any errors. Consider factors such as the criticality of the system, the prevalence of these surges, the nature of the failures (e.g. catastrophic failures or more benign feature degradation) and the budget constraints as guides for determining whether this matters or not.</p>

3.6.2 Long-Term Stability Test

Result	This test determines the ability of the system to continue functioning for a long duration, locating, for example, memory leaks that eventually cause system failures.
Purpose	Systems that function well in the short-term tests outlined above may still have unseen problems during long-term operation. The goal of this test is to determine the ability of the system to function for long periods of time under high traffic conditions. System availability continues as a critical measure in planning a network, as the fastest network means little if it constantly fails. Although this test suggests using a workload that resembles production traffic, it will also increase the traffic to a much higher level in order to accelerate the appearance of any stability issues.
Test specification	Connections
Constraints	Be sure that the connections/sec does not exceed the rate determined in test 3.5.2. Be sure that the total number of connections does not exceed the value determined in test 3.5.3. Keep bandwidth utilization below the amount determined in test 3.5.4.
Time estimate	Specification: about 30 minutes Run time: several days to weeks, depending on available time
Type of workload	If possible, use a workload that resembles a production network's traffic. Section 3.2 has information to help with this. Alternatively, use the workload from the Maximum Connections Test (3.5.3), which exercises the memory portion of the system under test more than most other tests.
Methodology	<ol style="list-style-type: none"> 1. If required, verify the proper functioning of the workload by following the steps in test 3.5.1. 2. Determine the maximum performance of the system under the chosen workload by following the steps in 3.5.3. Denote this maximum performance by MaxConn. 3. To keep the system running near the maximum, configure the test to ramp up over 600 seconds to a height of (MaxConn-5%). Use a large value for the step steady, dependent on the available time to run this test and on the minimum acceptable uptime. 4. Run the test. 5. At the beginning of the test, check that unexpected errors do not appear. 6. During the test, periodically check on the health of the system under test. Along with the system under test's own monitoring tools, use the test tool's real-time reporting for help in determining system health. Ensure traffic continues to flow, and that errors are not growing rapidly. 7. Keep running the test until the system under test fails or the test ends. 8. Analyze the results to determine whether system performance and behavior fits within the acceptable parameters.
What to look for	Timeouts, large numbers of open connections, and unexpected errors usually point to an overloaded system. For systems that support it, monitor the memory utilization. An increasing amount of memory utilization while under steady loads points to memory leaks that will eventually lead to system failure.

3.6.3 Failover Test

Result	This test determines the ability of the system to properly and adequately failover.
Purpose	Many network systems now use high-availability architectures that often use redundant systems. These systems usually fall into three categories, listed here in increasing level of failover protection: cold standby, warm standby, and dual-active. The best failover schemes completely preserve network activity and transactions, and are completely undetectable by users. Testing failover without network traffic does little to ensure that the network will remain viable and that users are minimally affected during and after system failover.
Test specification	Users
Constraints	Be sure that the connections/sec does not exceed the rate determined in test 3.5.2. Be sure that the total number of connections does not exceed the value determined in test 3.5.3. Keep the traffic under the level of simultaneous users found in test 3.5.5. Keep bandwidth utilization below the amount determined in test 3.5.4.
Time estimate	Specification: about 30 minutes Run time: about 1 hour
Type of workload	If possible, use a workload that resembles a production network's traffic. Section 3.2 has information to help with this. If the network will support stateful user applications (e.g. web site shopping carts, airfare searches, long FTP downloads), be sure that the workload incorporates these behaviors so that they can be tracked. Alternatively, use the workload from the Maximum Simultaneous Users Test, which maintains user connections long enough to determine whether they are affected.
Methodology	<ol style="list-style-type: none"> 1. If needed, verify the proper functioning of the workload by following the steps in test 3.5.1. 2. Determine the expected traffic level of the system, denoted here as ExpTrfc. If ExpTrfc cannot be determined, divide the maximum simultaneous users obtained from test 3.5.5 by 4 to achieve a baseline level of traffic. 3. Confirm that the redundant system under test will properly failover with no traffic. This can be done by confirming the failover settings and initiating a "failure" (e.g. pulling the cable that connects the system to the network). 4. Configure the test tool to ramp up to ExpTrfc over 240 seconds and then hold this amount steady for 600 seconds. 5. Start the test and wait for the 240-second ramp up period to complete. 6. Check to see that the traffic reaches steady state and that there are no failures. 7. Take note of the test time and initiate a failover. Usually, this is as simple as disconnecting the system from the network. 8. Let the test continue to run to completion. 9. Analyze the test results to determine the effect of the failover. The time noted in step 7 helps guide where/when to delve more deeply into the results. 10. Resolve any failover issues that have a fix or workaround, retesting to see if the new fixes improve the failover results.
What to look for	Application level errors, network timeouts and network retransmits point to failovers that may cause user-detectable errors. Often the most troublesome are application level errors in which user state is not maintained. This may cause a user to lose all the contents of their shopping cart.

4 Appendix

4.1 Test Phases

Avalanche contains a progressive load generation model that facilitates testing at different amounts of traffic. This simple facility allows users to configure Avalanche to hit certain amounts of traffic and hold that level for a fixed amount of time. During this step steady time, the user can then inspect the system under test to look for bottlenecks or other potential system issues. After each step steady time, Avalanche would then increase its load to the next step and hold, allowing testing to continue until the system under test reaches its limits.

- Ramp up: the beginning of load generation, where traffic is progressively and linearly increased to reach a particular load. This ramp up phase serves several purposes: to allow the system(s) being tested to “catch up” (e.g. allocate resources such as memory, connections, etc.) with the incoming traffic, to ensure that the system under test does not experience a sudden jump in traffic that causes it to fail or go into DoS defense mode, and to allow the traffic to hit a certain initial value before continuing with more traffic.
- Step steady time: the amount of time to maintain a certain level of traffic after ramp up or a step.
- Step ramp time: the amount of time to take to increase to the next level of traffic.
- Step height: an increase in traffic to a new level.

4.2 Things to Look for During System Failures

When a system becomes overloaded, look for one or more of the following behaviors on the system:

- Timeouts for all new incoming connections
- Resets of incoming connections and while continuing to process current connections
- Accepts and holds open incoming connections but refuses to forward traffic
- A complete lockup/failure

Often, system overloads and failures may be seen in the statistics:

- A topping out of open connections
- A drop in open connections
- Increase in time to TCP SYN/ACK
- Topping out of bandwidth
- Decrease in bandwidth

4.3 Spirent Communications Products

Spirent Communications is a worldwide provider of integrated performance analysis and service assurance systems for next-generation network technologies. Our solutions accelerate the profitable development and deployment of network equipment and services by emulating real-world conditions in the lab and assuring end-to-end performance of large-scale networks. Full details of our award-winning products can be found at <http://www.spirentcom.com/L4-7/>.

4.3.1 Avalanche

The Avalanche family of capacity assessment products provide high-performance protocol-accurate L4-L7 stress testing under high loads, ensuring that your device, application or infrastructure will excel under real-world conditions. With support for protocols such as http, SSL, RTSP/RTP (QuickTime and Real Networks), MMS, FTP, SMTP and POP3, Avalanche is the world's first system to emulate the full user/browser experience. As a result, Avalanche gives you insight into your infrastructure's architectural effectiveness, points of failure, modes of performance degradation, robustness under critical load, and potential performance bottlenecks. In other words, Avalanche lets you see what your users will see, and tune your equipment and infrastructure accordingly. With Avalanche, you can deploy your products and services with confidence.

4.3.2 Reflector

Until recently, network equipment manufacturers and service providers only had one real option for assessing their products' capacity to withstand high-volume Internet traffic: they had to build complex application-layer infrastructures in the lab using large numbers of servers. Spirent offers a better way. The Reflector family of capacity assessment products realistically simulates the behavior of large Web, application and data server environments. Combined with a capacity assessment solution such as Spirent's Avalanche, this cost-effective and intuitive system provides a total solution for simulating the world's largest Web infrastructures. By generating accurate and consistent HTTP responses to Avalanche's high volume of realistic Internet user requests, Reflector tests to capacity any equipment or network connected between the two systems.

5 Acknowledgements

I would like to thank the following people for providing feedback and assistance during the creation of this document:

- Roy Chua
- Alan Newman
- Sanjay Raja

6 References

- [CAIDAO2] University of California at San Diego, C Shannon, D Moore, and K.C. Claffy
Beyond Folklore: Observations on Fragmented Traffic
IEEE/ACM Transactions on Networking, 709-720, December 2002.
- [WEBPDAO3] Internet.com Webopedia
Online Dictionary
<http://www.webopedia.com/>

(Footnotes)

¹HTTP 1.0 was modified after its inception to have a persistent mode as well, called keep-alive, but its use is less prevalent.